ADAPTIVE ACCESSING METHOD AND SYSTEM FOR SINGLE LEVEL STRONGLY CONSISTENT CACHE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the technical field of data consistency in cache and, more particularly, to an adaptive accessing method and system for single level strongly consistent cache.

2. Description of Related Art

Because terminal devices have become diversified and network interconnections have been widely used, the application in integrating wireless communication and the Internet generally requires a high system performance. For coping with this, cache has been proposed as means for improving system performance. As such, cache has been widely used in the current Internet application service for increasing the transmission performance of the system. As to the field of wireless communication, cache is also critical due to limited bandwidth thereof.

In the application of cache in wireless or wired communications, a consistency between both parties involving in the data communication is the most important consideration in the Internet service. In current techniques about data consistency in cache, there are two most widely used strongly consistent algorithms: the poll-each-read and callback. Referring to FIG. 1, a network communication structure according to prior art is shown. In the implementation of poll-each-read algorithm, whenever accessing a cached data entry, the client 11 has to poll the server 12 about whether the cached data entry is valid. If yes, the server

12 responses a validation affirmation. Otherwise, the server 12 sends the latest cached data entry to the user 11. In practice, the server 12 maintains a valid bit pointer c_v for each user 11 having the cached data entry and performs the following algorithm:

Algorithm I. Poll-Each-Read.

I.1. Entry Update (Server): When a cached data entry is updated, for every client that has the cached data entry, the server sets c_v to 0, wherein "0" implies that the cached data entry is invalidated.

I.2. Entry Access (Client): To access a cached data entry, a client sends an entry access message to the server. The message contains an access type bit c_a. If the client does not have a cached data entry (either the entry is first accessed or was replaced), then c_a is 1. In this case, the cached data entry in the server should be sent to the client. If the client has the cached data entry, then c_a is set to 0. In this case, the cached data entry should be validated by the server.

I.3. Entry Access (Server): The server receives a cached data entry access message from a client. Let c_v be the validation bit for that client.

I.3.1. If the client does not have the cached data entry (i.e., $c_a = 1$), the server sends the entry to the client, and c_v is set to 1.

I.3.2. If $c_a = 0$ and $c_v = 0$, then the server sends the cached data entry to the client. The bit c_v is set to 1.

I.3.3. If $c_a = 0$and $c_v = 1$ then the server returns validation affirmation to the client.

Referring to FIG. 2, an example of executing the poll-each-read algorithm is shown. At time t0, the client 11 desires to access a data entry not present in the cache thereof. Thus, the client 11 send a request having an access type bit of one (i.e., c_a=1) to the server 12. When receiving the request, the server 12 sends the cached data entry to client 11 and set c_v to one. At time t1, the client 11 desires to access an entry present in the cache thereof. Thus, the client 11 sends a request having an access type bit of zero (i.e., c_a=0) to the server 12. When receiving the request, the server 12 checks that the c_v is one and thus responses a validation affirmation to the client 11. As such, the client 11 can directly access the cached data entry in the cache. At time t2, the server 12 updates a cached data entry in the cache thereof and sets c_v to zero. At time t3, the client 11 desires to access an entry present in the cache thereof. Thus, the client 11 sends a request having an access type bit of zero (i.e., c_a=0) to the server 12. When receiving the request, the server 12 sends the cached data entry to the client 11 and sets c_v to one.

In the implementation of callback algorithm, once a cached data entry in the server 12 is updated, the server 12 informs the client 11 to set the cached data entry to be invalid. In practice, the server 12 maintains a

valid bit pointer c_v for each user 11 having the cached data entry and performs the following algorithm:

Algorithm II. Callback.

II.1.Entry Update (Server): When an update occurs, for every client that has the cached data entry, if $c_v = 1$, the server sends an invalidation message to the client. Then the server sets c_v to 0.

II.2.Entry Update (Client): When the client receives the invalidation message, the cached data entry is invalidated and the storage can be reclaimed to cache another data entry. The client sends an acknowledgement message to the server.

II.3.Entry Access (Client): If the cached data entry exists, then the client uses the cached data entry. Otherwise, the client sends an entry access message to the server. Eventually, the client will receive the cached data entry from the server.

II.4.Entry Access (Server). When the server receives an entry access message from a client, it sends the cached data entry to the client. Let c_v be the validation bit for that client. The server sets c_v to 1.

Referring to FIG. 3, an example of executing the callback algorithm is shown. At time t0, the client 11 desires to access a data entry not present in cache thereof. Thus, the client 11 sends a cached data entry

request to the server 12. When receiving the request, the server 12 sends the cached data entry to the client 11, and sets c_v to one. At time t1, the client 11 can directly access the cached data entry in its cache. At time t2, the server 12 updates the cached data entry, and sends an invalidate message to the client 11 since c_v is one. Then, the server 12 sets c_v to zero. In response, the client 11 sends an acknowledgement. At time t3 and t4, the server 12 updates the cached data entry. At time t5, the client 11 desires to access the invalid or not-existent data entry in the cache. Thus, the client 11 sends a data entry access request to the server 12. When receiving the request, the server 12 sends the cached data entry to the client 11 and sets c_v to one.

In view of above algorithms, it is found that, when the update frequency of the server 12 is low, the client 11 still has to poll the server 12 for accessing non-updated cached data entry in using the poll-each-read algorithm. This inevitably wastes a lot of bandwidths. On the contrary, if the update frequency of the server 12 is high, the server 12 still continuously sends invalidate message to the client 11 in using the callback algorithm even when the client 11 does not access data. This also inevitably wastes a lot of bandwidths. Therefore, it is desirable to provide a novel system and method therefor to mitigate and/or obviate the aforementioned problems.

SUMMARY OF THE INVENTION

The object of the present invention is to provide an adaptive accessing method and system for single level strongly consistent cache, capable of dynamically selecting a poll-each-read algorithm or a callback

algorithm based on the update frequency of the server and the access frequency of the client for effectively reducing the communication cost.

In accordance with one aspect of the present invention, there is provided an adaptive accessing system for single level strongly consistent cache. A server is provided to have a cache, at least one cached data entry, and a first counter and a second counter corresponding to each client of each cached data entry. The first counter measures the number of cycles in an observed period, and the second counter measures the number of cycles that have updates in the cycles, wherein a cycle is defined as a period between two consecutive data accesses. At least one client is provided to connect to the server via a communication link, and each client has a cache. A dynamic adjustment module corresponding to each client of each cached data entry is provided for selecting a poll-each-read algorithm or a callback algorithm based on a ratio of the first counter and the second counter to maintain a consistency of the caches in the client and the server.

In accordance with another aspect of the present invention, there is provided an adaptive accessing method for single level strongly consistent cache. First, in the server, a first counter is used for measuring the number of cycles in an observed period, and a second counter is used for measuring the number of cycles that have updates in the cycles, wherein a cycle is defined as a period between two consecutive data accesses. Next, there is determined a ratio of the first counter and the second counter. Then, there is selected a poll-each-read algorithm or a callback algorithm based on the ratio.

Other objects, advantages, and novel features of the invention will become more apparent from the detailed description when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing a network communication structure using cache technique;

FIG. 2 schematically illustrates an implementation of poll-each-read algorithm;

FIG. 3 schematically illustrates an implementation of callback algorithm;

FIG. 4 is a block diagram of the adaptive accessing system for single level strongly consistent cache in accordance with the present invention; and

FIG. 5 shows a comparison of communication cost among the present method, the poll-each-read algorithm and the callback algorithm.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference to FIG. 4, there is shown a network communication structure according to the adaptive accessing method and system for single level strongly consistent cache of the present invention. As shown, a server 42 is connected to at least one client 41 via a wired or wireless communication link. The server 42 and the client 41 are provided with caches 421 and 411, respectively, for enhancing the transmission performance of the system. Furthermore, an adaptive adjustment module 43 is provided for selecting the poll-each-read algorithm or callback algorithm to maintain the data consistency of the cache. When being

applied to a wireless network environment designed by WAP (wireless application protocol), the client 41 is a mobile device and the server 42 is a WAP gateway.

The dynamic adjustment module 43 is configured to maintain a consistency of cache with a minimum cost. For obtaining the time at which the dynamic adjustment module 43 selects poll-each-read or callback algorithm, it is assumed that α is the probability of at least one data update occurred between two data accesses; x is cost of sending a request , response, or message indicating whether cached data entry to be accessed is valid or not; and y is cost of transmitting the complete updated data entry. Both x and y are measured in terms of bit. In the poll-each-read algorithm, the cost of step I.2 is x; the probability of step I.3.2 is α and its cost is α y; and the probability of step I.3.3 is α and its cost is $(1-\alpha)$x. Hence, the communication cost of each data access in poll-each-read algorithm can be expressed as follows:

$$C_I = x+\alpha y+(1-\alpha)x = \alpha(y-x)+2x \qquad (1)$$

In the callback algorithm, the steps II.1 to II.4 are executed only when data is updated, which has a probability of α. Thus, the total cost of steps II.1 and II.2 is 2α x; the cost of step II.3 is α x; and the cost of step II.4 is α y. Hence, the communication cost of each data access in the callback algorithm can be expressed as follows:

$$C_{II} = 2\alpha x+\alpha x+\alpha y = \alpha(3x+y) \qquad (2)$$

From equations (1) and (2), it is determined a condition for selecting the poll-each-read or callback algorithm as follows:

$$C_I > C_{II} \Leftrightarrow \alpha(y-x) + 2x > \alpha(3x+y)$$

$\Leftrightarrow 2x > 4\alpha x$

$\Leftrightarrow \alpha < 1/2$ (3)

That is, when α <1/2, the communication cost of using the callback algorithm is less. On the contrary, when α >1/2, the communication cost of using the poll-each-read algorithm is less.

In order to determine the value of α, a cycle is defined as a period between two consecutive data accesses. The server 42 is associated with two counters n_u and n_c for each cached data entry, wherein the counter n_u measures the number of the cycles that have updates in the cycles, and, the counter n_c measures the number of the cycles in an observed period. Hence, the probability of at least one data update occurred between two data accesses is equal to the ratio of n_u and n_c, i.e., $\alpha = n_u/n_c$. These two counters n_u and n_c are operating as follows:

1. In using the poll-each-read algorithm, if the server 42 receives an cached data entry access request from user 41 (step I.3), n_c is incremented. If the client 41 desires to access a cached data entry existed in the cache (i.e., c_a=0), and the server 42 has received the cached data entry access request from the client 41 and the cached data entry is invalid (i.e., c_v=0) (step I.3.2), n_u is incremented.

2. In using the callback algorithm, each cached data entry in the client 41 is associated with a third counter n_c* for measuring the number of accesses since the previous update. When the client 41 accesses a cached data entry in the cache (step II.3), n_c* in the client 41 incremented. When the server 42 updates a cached data entry (step II.1), n_u is incremented. If a cached data entry in the client 41 is set to be invalid

(step II.2), the client 41 sends n_c* to the server 42, and sets n_c* to be zero. The server 42 adds n_c* to n_c.

3. When n_c is greater than a predetermined value Nc, the server 42 determines the value of α by $\alpha = n_u/n_c$, and, based on equation (3), it is determined whether to use the poll-each-read algorithm or the callback algorithm. Afterwards, both n_u and n_c are set to zero.

In view of above, by observing a change of α, the present invention can dynamically select the poll-each-read or callback algorithm for maintaining a data consistency of cache, thereby reducing the communication cost of the system to a minimum. With reference to FIG. 5, there is shown a comparison of communication cost among the present method, the poll-each-read algorithm and the callback algorithm under a condition of Nc=10, where μ is the counting rate of update event; λ is the counting rate of access event; and y=10x. As shown, the present method can provide a better performance.

Although the present invention has been explained in relation to its preferred embodiment, it is to be understood that many other possible modifications and variations can be made without departing from the spirit and scope of the invention as hereinafter claimed.